BioDCASE 2026 Task 3 Technical Report: Mid-Frequency Log-Mel BalancedDSCNN Ensembles for Tiny Bird Audio Recognition

Yuhuan You
Peking University
2000017809@stu.pku.edu.cn

Abstract

This report describes four submitted systems for BioDCASE 2026 Task 3, Bioacoustics for Tiny Hardware. All submitted systems use the official three-second mono audio inference interface, a log-mel frontend, PyTorch inference checkpoints, and same-name INT8 TFLite models. The main design is a compact depthwise-separable convolutional network, BalancedDSCNN, trained with a mid-frequency log-mel representation. The primary hidden-test system is a Train+Validation three-member weighted ensemble, while the package also includes a smaller single-model Train+Validation system, a four-member Train+Validation backup ensemble, and a clean public-validation ensemble for auditability. External resources used during training are disclosed explicitly: BirdNET V2.4 was used only as a soft-label teacher, and a small filtered Xeno-canto set of public target-species recordings uploaded before 2025-04-01 was used as low-weight regularization. No external model is required at inference time.

1 System Overview

The task input is a three-second audio clip resampled to 24 kHz. Each submitted system applies a log-mel frontend with a 2048-sample analysis window, 512-sample stride, 64 mel channels, and a 500–9000 Hz band. This configuration was selected after validation experiments showed that it retained more species-discriminative high-frequency detail than the official baseline frontend while remaining compatible with the provided inference handler and generated TFLite path.

The neural architecture is BalancedDSCNN, a depthwise-separable convolutional classifier. The model uses an initial two-dimensional convolution, separable convolution blocks, intermediate max pooling, global average pooling, dropout, and a final linear classifier over the eleven Task 3 labels. Ensemble submissions wrap several BalancedDSCNN members in a single PyTorch module and return the log of a fixed weighted probability average. The same ensemble module is converted to a same-name TFLite file, so each submitted solution contains both a host inference checkpoint and an embedded-model artifact.

2 Training Procedure

The final-lane systems were trained on the official training and validation audio, because the hidden evaluation set is separate and the final package should use all available labeled development examples once model selection is complete. The clean audit system was selected using the public validation split and is included to make the validation-side performance trail explicit.

Training used cross-entropy with weak soft-label distillation from BirdNET V2.4. BirdNET was used as a training-time teacher only; it is not loaded, called, or packaged for inference. Low-weight Xeno-canto target-species clips were used as regularization examples after filtering by upload date, license, target species, and audio validity. The external audio weight was deliberately kept low to reduce the risk of domain mismatch dominating the official development data.

3 Submitted Solutions

Folder	System	PTH bytes	TFLite bytes	TFLite MACs	TFLite params
You_PKU_task3_1	TrainVal three-member low-8423 ensemble	923225	894364	121567233	197352
You_PKU_task3_2	TrainVal single BalancedDSCNN seed 8414	307962	293912	40522400	65784
You_PKU_task3_3	TrainVal four-member low-8424 backup ensemble	1232126	1192924	162089644	263136
You_PKU_task3_4	Clean validation pruned-14 ensemble	4286089	4178756	567313754	920976

Table 1: Submitted systems and local submission-test resource measurements. Embedded board timing and RAM are left as N/A because no ESP32-S3-Korvo-2 board measurement was available in this environment.

All four submitted solutions passed the official repository's local `submission_test.py` smoke test in both PyTorch and TFLite modes on the bundled eleven-file test set. The smoke set is an execution check rather than a hidden-test performance estimate. The clean public-validation model selection evidence favored the pruned-14 Mid-feature ensemble, which reached 0.8197 Top-1 and 0.9752 ROC AUC on the public validation split. The Train+Validation systems intentionally do not provide a clean validation score because they use all official development labels.

4 External Resources and Disclosure

The submitted inference pipeline is self-contained and does not require any external service, pretrained teacher, or external audio file at test time. The training workflow used BirdNET V2.4 as a soft-label teacher. It also used a small Xeno-canto set of target-species recordings filtered to public uploads before 2025-04-01, excluding NoDerivatives licenses and invalid audio. The filtered external clips were used at low sample weight together with official development examples. This report discloses these resources because BioDCASE rules require public external datasets and trained models to be disclosed.

Generative AI coding assistants, including ChatGPT/Codex, were used for code inspection, experiment orchestration, packaging checks, report drafting, and submission-preparation review. Final system choices, data-use boundaries, and package contents were supervised by the author. No hidden evaluation labels were accessed or inferred.

5 Reproducibility Notes

Each solution folder contains its own `config_submission.yaml`, `submission_test.py`, `your_submission_model/label` a PyTorch model, a same-name TFLite model, and replacement PyTorch model code under `pipeline_pytorch/`. The replacement code defines the BalancedDSCNN and BalancedDSCNNWeightedEnsemble classes required by the submitted configs. The model frontend remains the official log-mel path instantiated through the provided `FeatureHandler`.

6 References

BioDCASE 2026 Task 3: `https://biodcase.github.io/challenge2026/task3`.
BioDCASE-Tiny 2026 baseline repository: `https://github.com/birdnet-team/BioDCASE-Tiny-2026`.
BioDCASE 2026 submission instructions: `https://biodcase.github.io/challenge2026/submission`.
Xeno-canto public bird sound archive: `https://xeno-canto.org/`.
BirdNET Analyzer: `https://github.com/birdnet-team/BirdNET-Analyzer`.